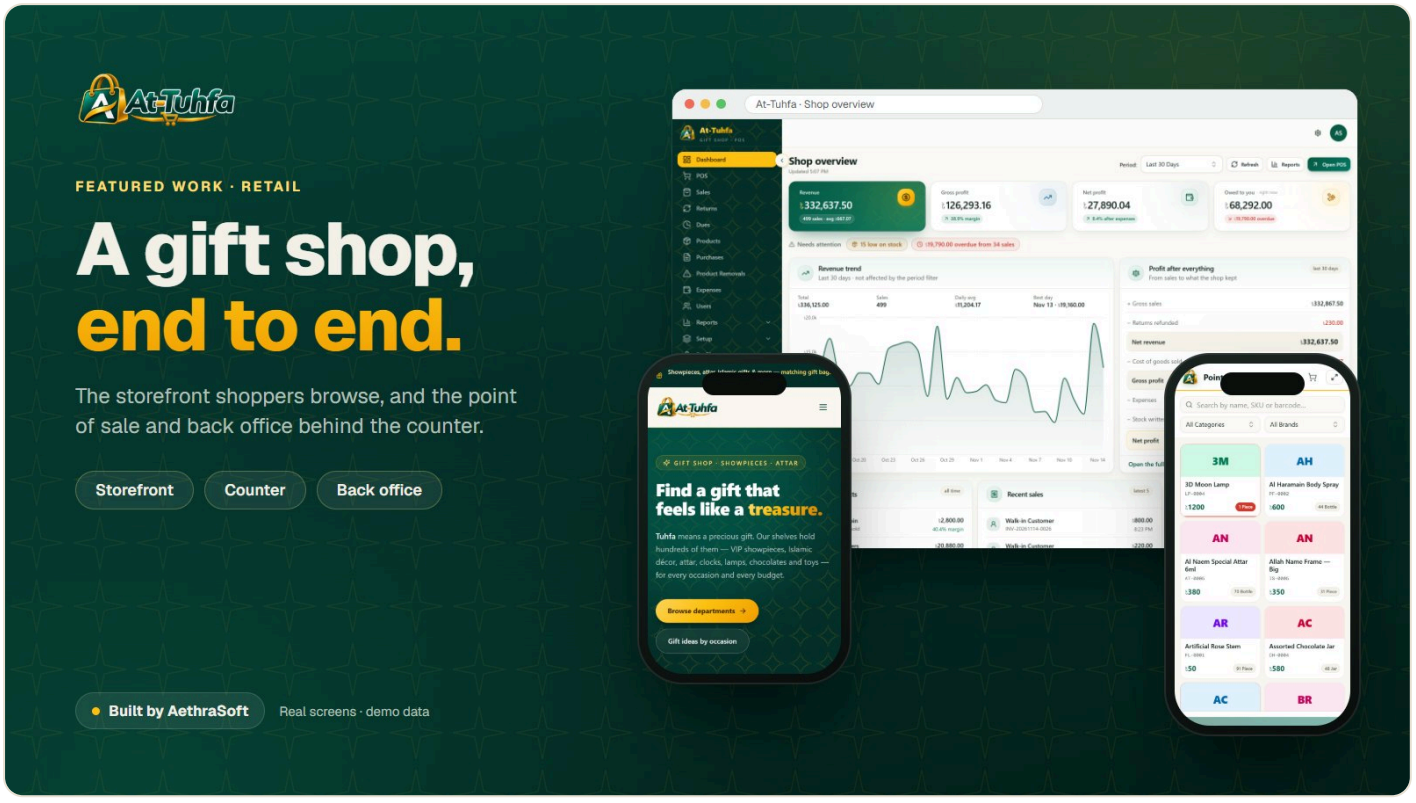
GIFT RETAIL · STOREFRONT · POINT OF SALE

At-Tuhfa

A gift shop's storefront and the point-of-sale system behind its counter — selling, stock, credit, returns, expenses and a profit & loss statement that actually closes. Built end to end by AethraSoft.

At-Tuhfa is a gift shop in Bangladesh: showpieces, attar, Islamic gifts, clocks, lamps and chocolates, with a gift bag to go with every present. It has two very different audiences. Shoppers want to know what is on the shelves and what it costs. The people behind the counter want to sell quickly, keep the stock honest, chase what customers owe and know, at the end of the month, what the shop actually earned. This project serves both from one codebase.



The storefront and the back office, side by side

At a glance

17 back-office screens, plus the public storefront	**16** backend service modules	**18** database models	**6** roles on an editable permission grid
26 selling units, whole and fractional	**2** languages of end-user documentation		

> **NOTE** Every back-office screen on this page comes from a running instance loaded with a seeded demonstration shop: the At-Tuhfa catalogue and prices, with about 1,350 invoices across 90 days. Customers, staff and every figure are invented, not the shop's real trading. The storefront uses illustrative stock photography.

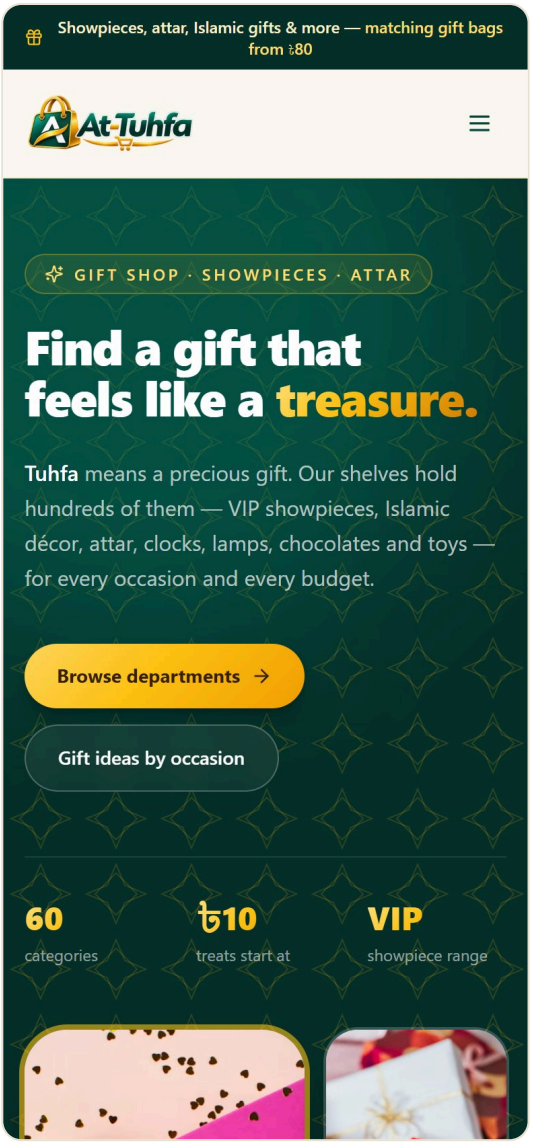
Two audiences, one project

The storefront and the point of sale are one Next.js and NestJS project, served by **one Node process on one port**. A small piece of middleware sends the API, uploads and API docs to NestJS and everything else to Next.js. One deployment, one certificate, one log to read.

- **For shoppers** — a fast, fully static storefront for discovering gifts by department, occasion and budget, and for deciding to visit the shop.
- **For the shop** — a point of sale, inventory, credit, returns, expenses, reports, user roles and an audit trail, behind a sign-in.

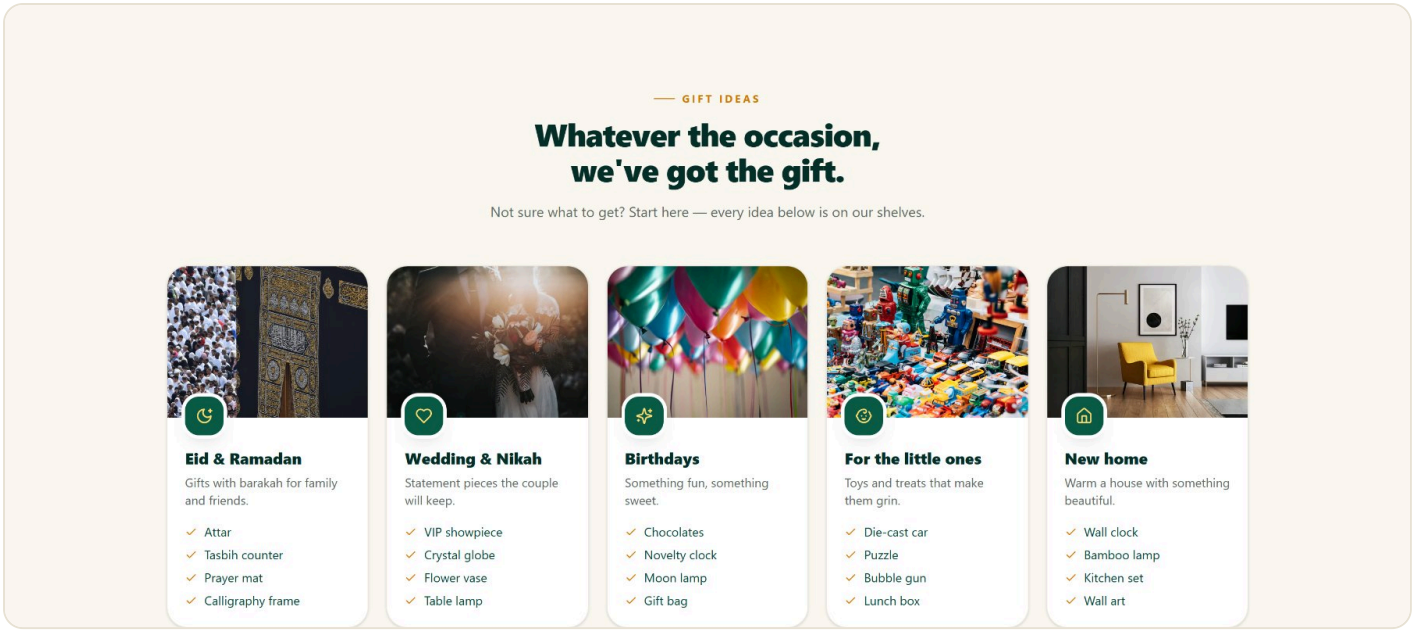
For shoppers: the storefront

The storefront answers the questions a gift buyer actually has. What do you have? What would suit a wedding, an Eid visit or a new baby? What can I get for ৳500? It opens on the shop's own promise, with the departments one tap away, and it is built phone-first, because that is where most shoppers will see it.



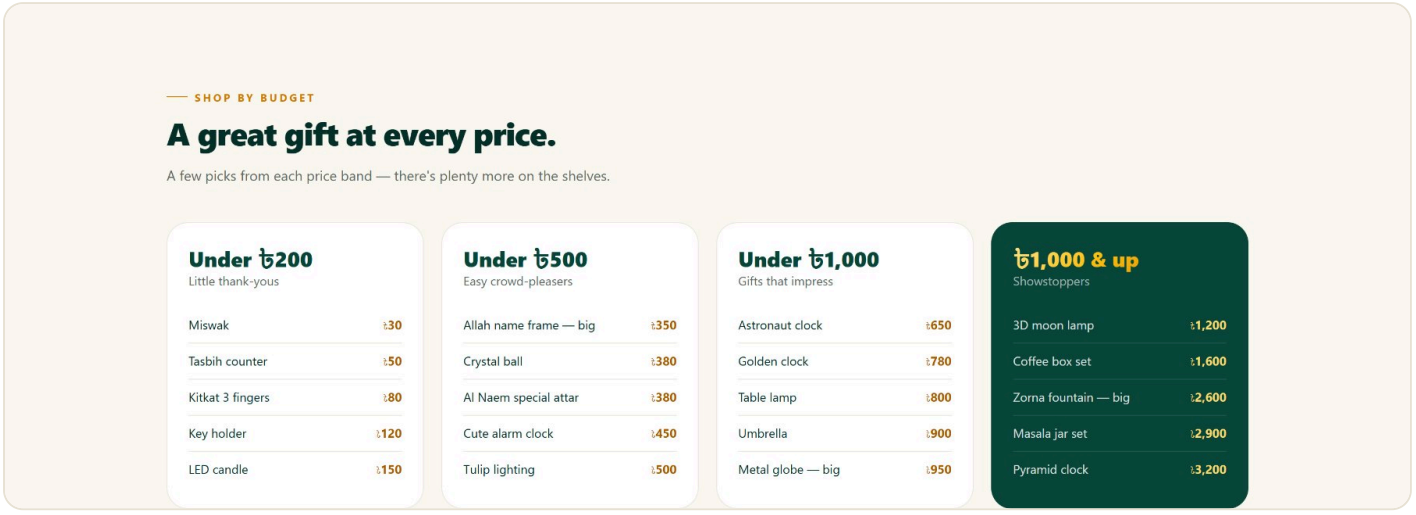
The storefront on a phone

Departments group the shop's POS categories the way a shopper thinks about them, each with a "from" price, so nobody has to guess whether a section is within budget. For the undecided, gift ideas are grouped by occasion.



Gift ideas by occasion

Budget bands do the rest of the browsing, and the Islamic corner gets a section of its own. Matching gift bags start at ৳80 at the counter.



Gifts by budget

The storefront is deliberately simple: it is for discovery and a visit to the shop. There is no cart, no online payment and no delivery. Every word and price lives in one data file, so the page is fully static and keeps working even if the back office is down.

The problem behind the counter

A shop owner can tell you today's takings. Ask what the shop *earned* and the answer gets vague, because the honest version needs four things at once: what the goods cost, what went out on credit,

what came back or broke, and what the month's overheads were. Those four numbers usually live in four different places: a cash drawer, a notebook, somebody's memory and a shoebox of bills.

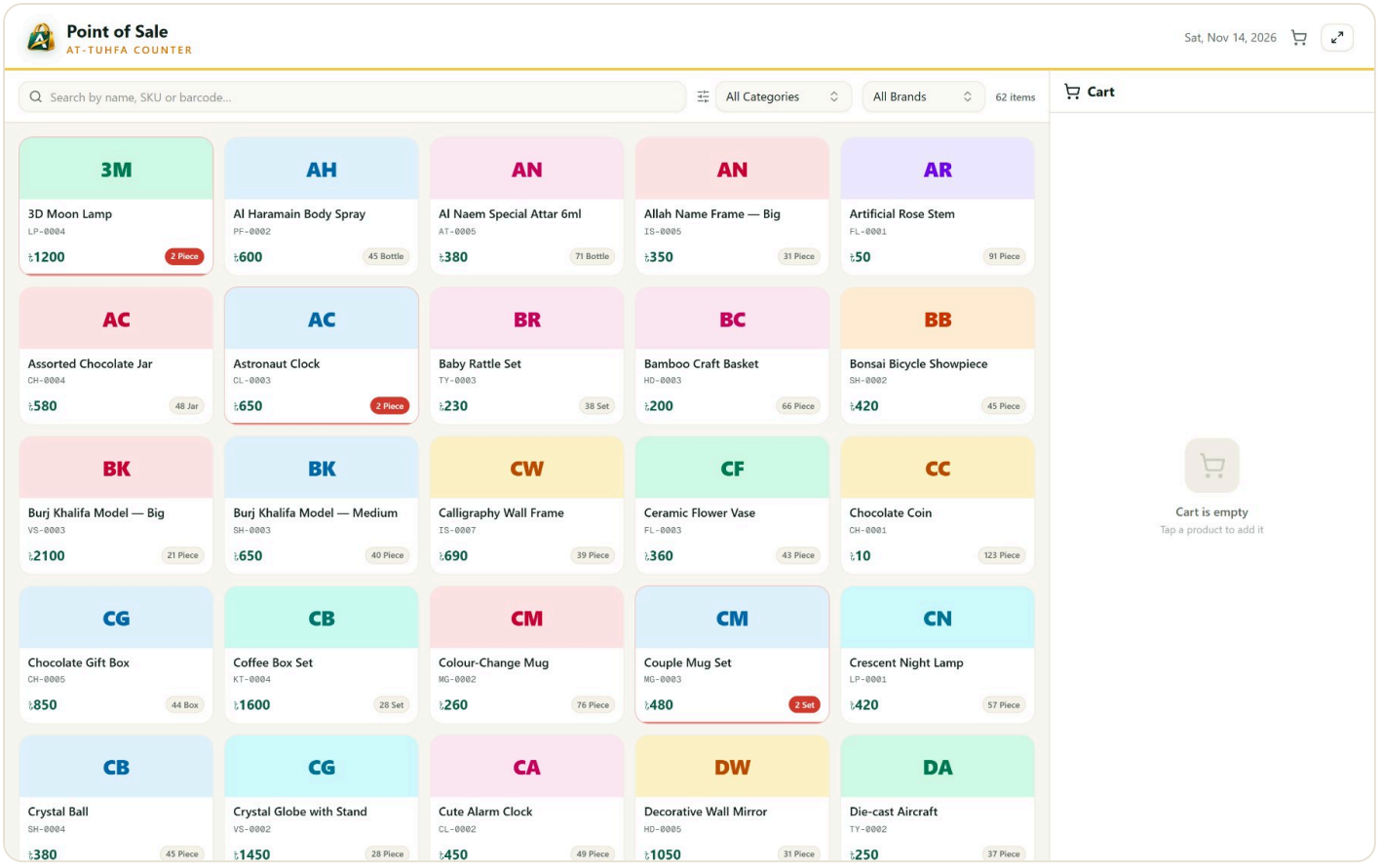
At-Tuhfa's back office keeps them in one system and does the subtraction:

```
Gross sales
  − returns           → net revenue
  − cost of goods     → gross profit
  − expenses          → operating profit
  − stock written off → net profit
```

That last line is the one the owner actually wanted, and it is on the screen without anybody reaching for a calculator.

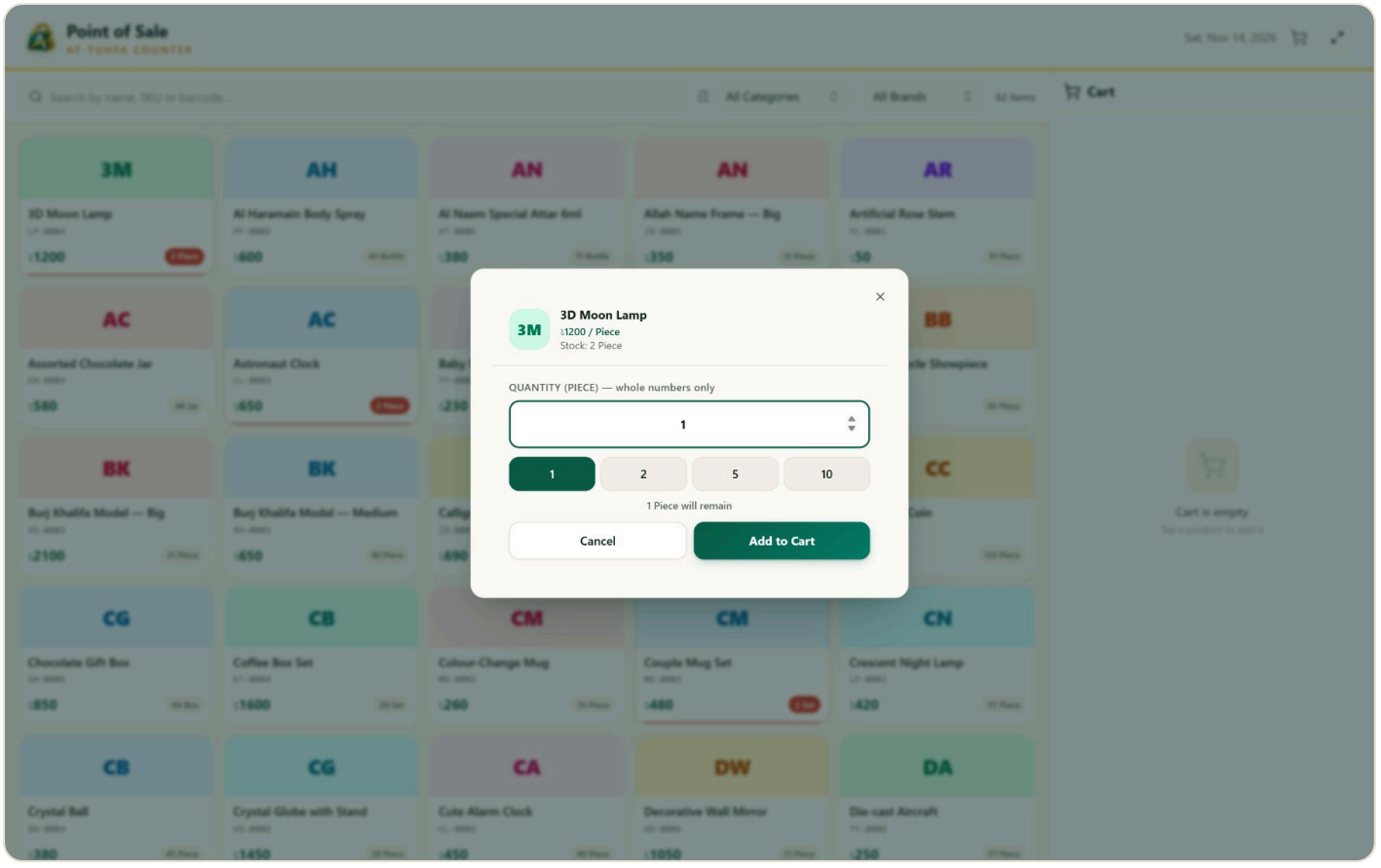
At the counter

The register is deliberately its own world: no sidebar, no dashboard chrome, nothing to click by accident during a queue. Search by name, SKU or barcode. A hardware scanner types into the same box, and the grid narrows to the matching product. Products that are out of stock never appear, so a cashier cannot sell what isn't there.



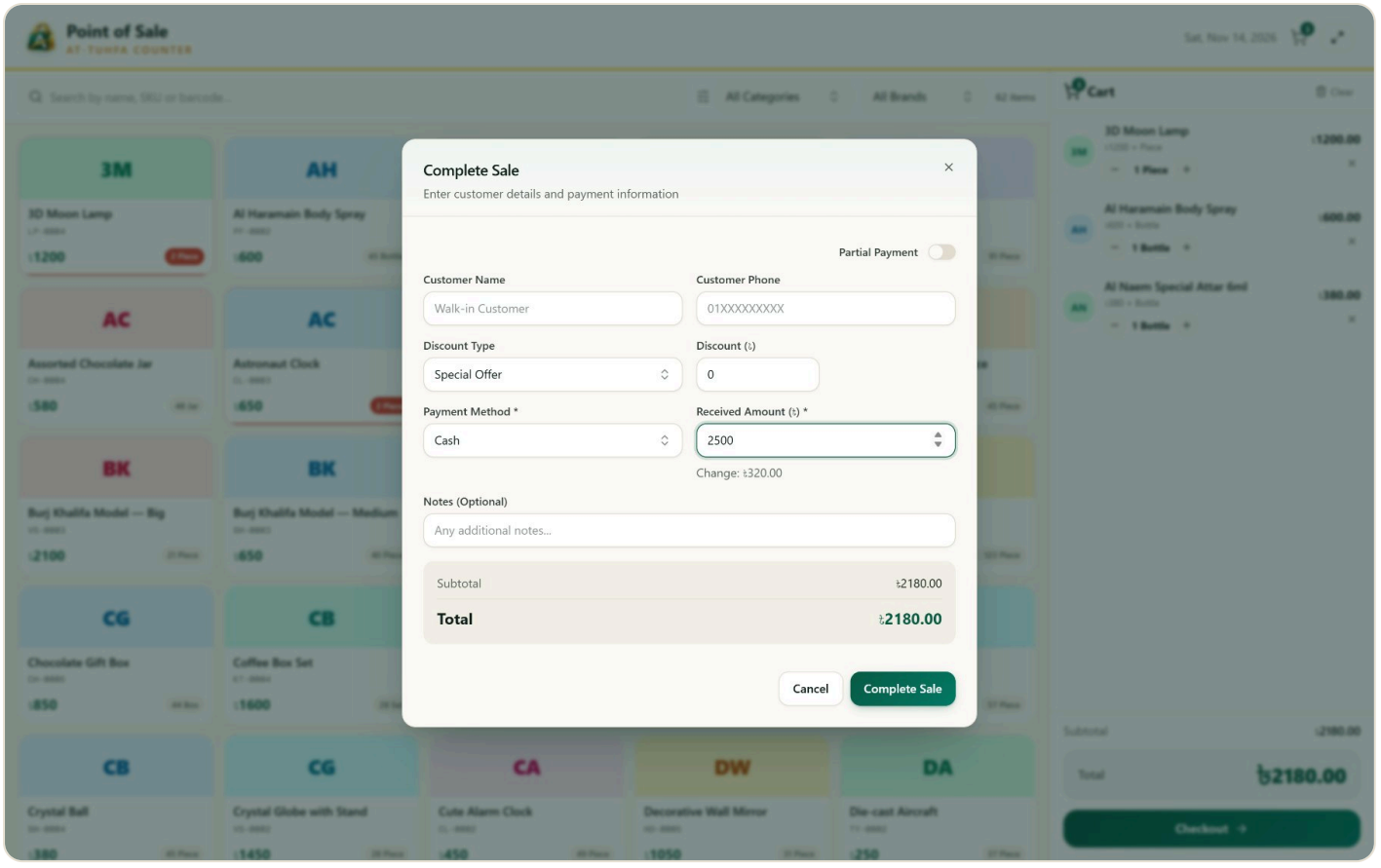
The POS terminal

Quantity is unit-aware. Nineteen units are whole-number only, so you cannot sell 1.5 clocks. Seven are decimal, which is how attar sells by the millilitre.



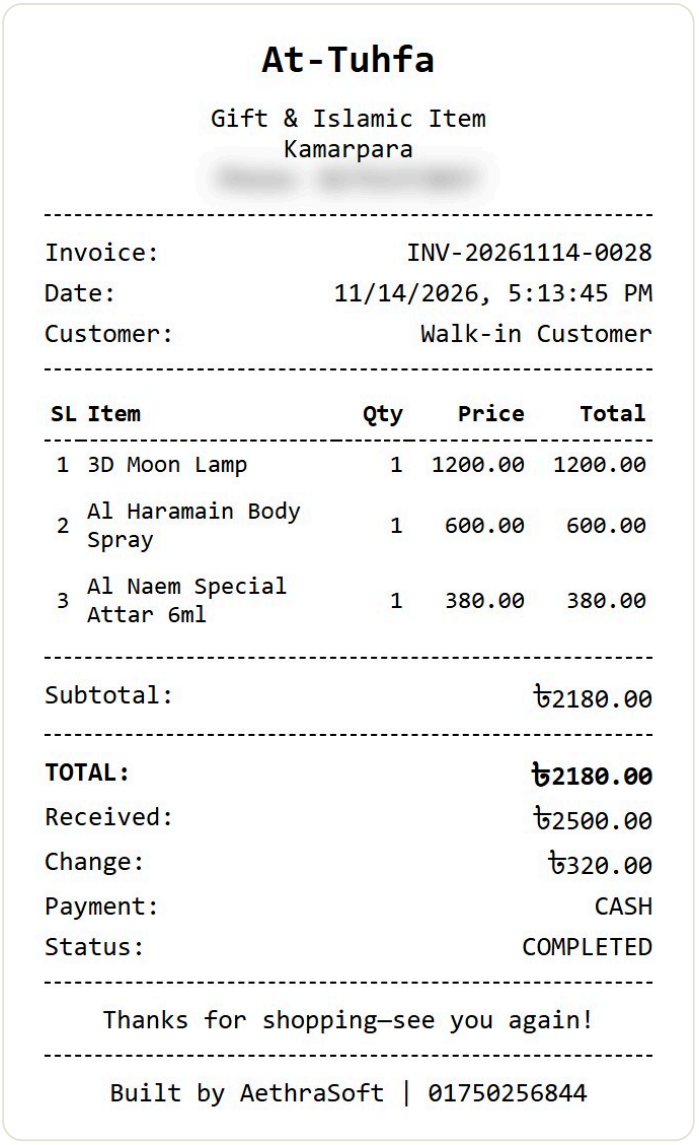
Choosing a quantity

Checkout takes customer details, a discount (a fixed amount, a percentage or a special-offer price), one of five payment methods (cash, card, mobile banking, bank transfer or a digital wallet) and the cash handed over, then works out the change. Flip **Partial Payment** and the balance becomes a due against the customer's phone number, which is the identity the whole credit system is keyed on.



Completing a sale

The receipt prints the moment the sale completes, with an invoice number from a readable `INV-YYYYMMDD-####` sequence.



At-Tuhfa

Gift & Islamic Item
Kamarpara
[illegible]

Invoice: INV-20261114-0028
Date: 11/14/2026, 5:13:45 PM
Customer: Walk-in Customer

SL	Item	Qty	Price	Total
1	3D Moon Lamp	1	1200.00	1200.00
2	Al Haramain Body Spray	1	600.00	600.00
3	Al Naem Special Attar 6ml	1	380.00	380.00

Subtotal: ৳2180.00

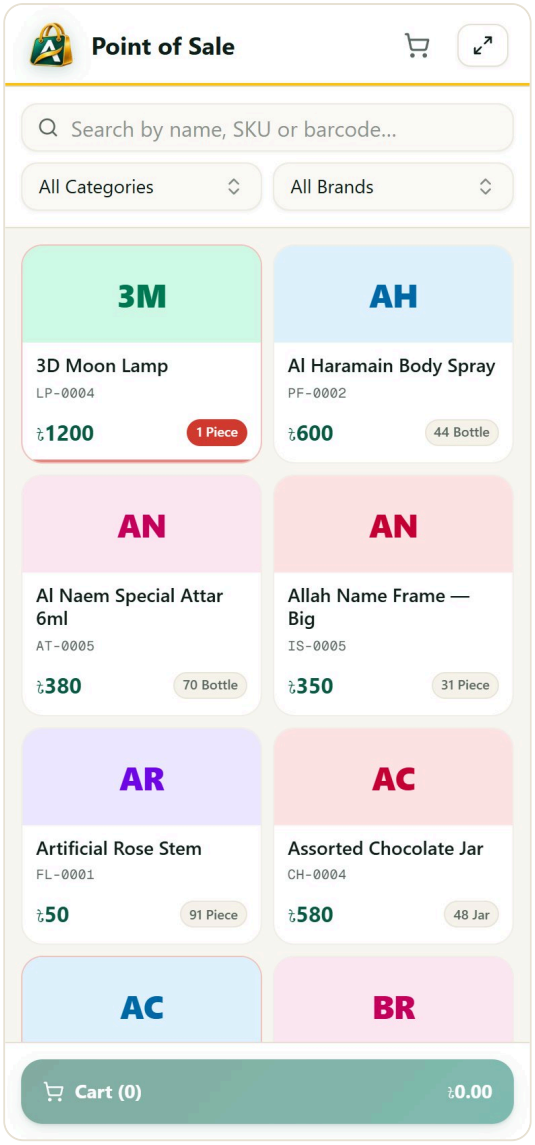
TOTAL: ৳2180.00
Received: ৳2500.00
Change: ৳320.00
Payment: CASH
Status: COMPLETED

Thanks for shopping—see you again!

Built by AethraSoft | 01750256844

The printed receipt (shop phone blurred)

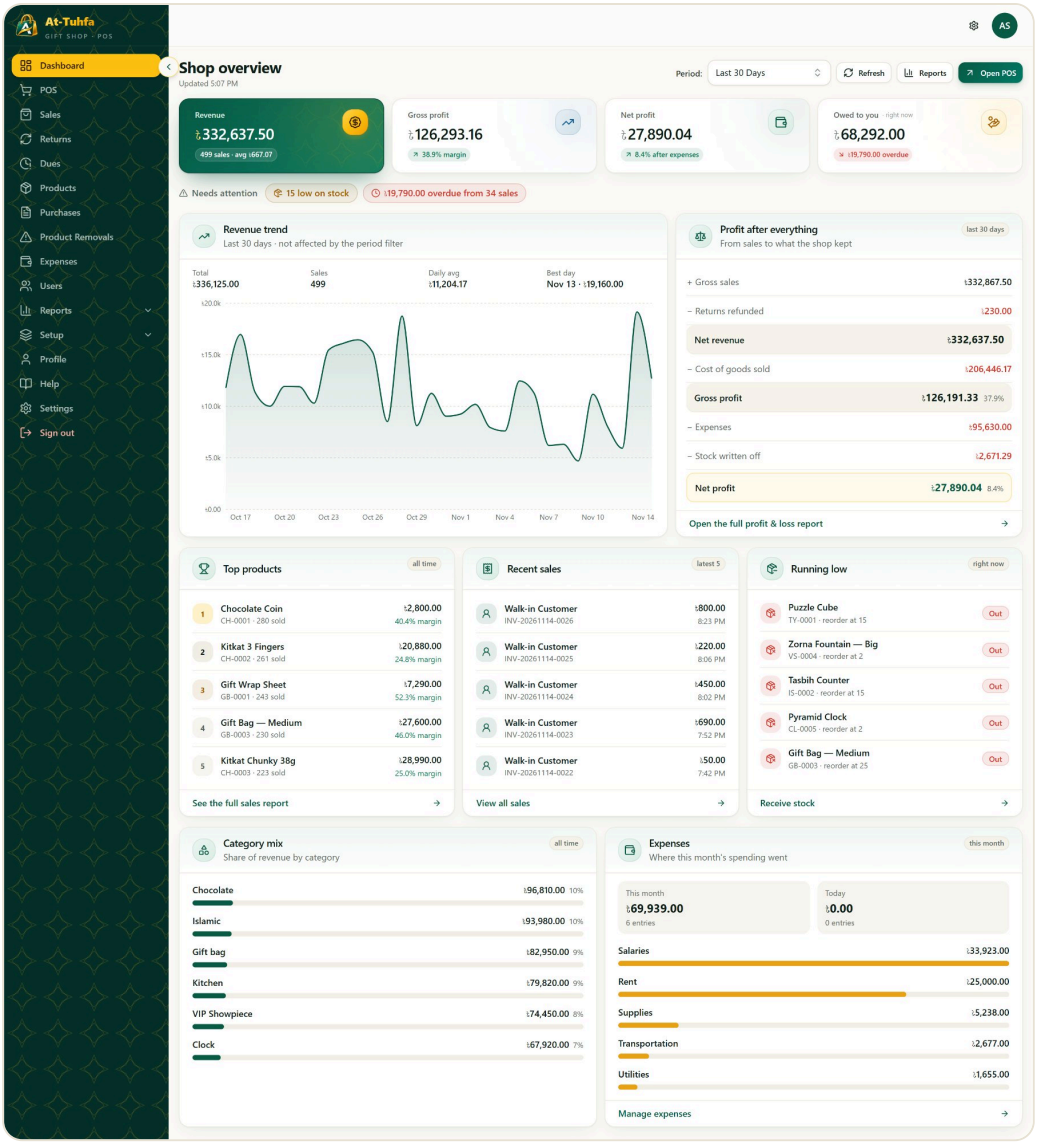
On a phone the grid reflows and the cart becomes a bottom sheet, because plenty of shops run the counter from a handset.



POS on a phone

The morning look

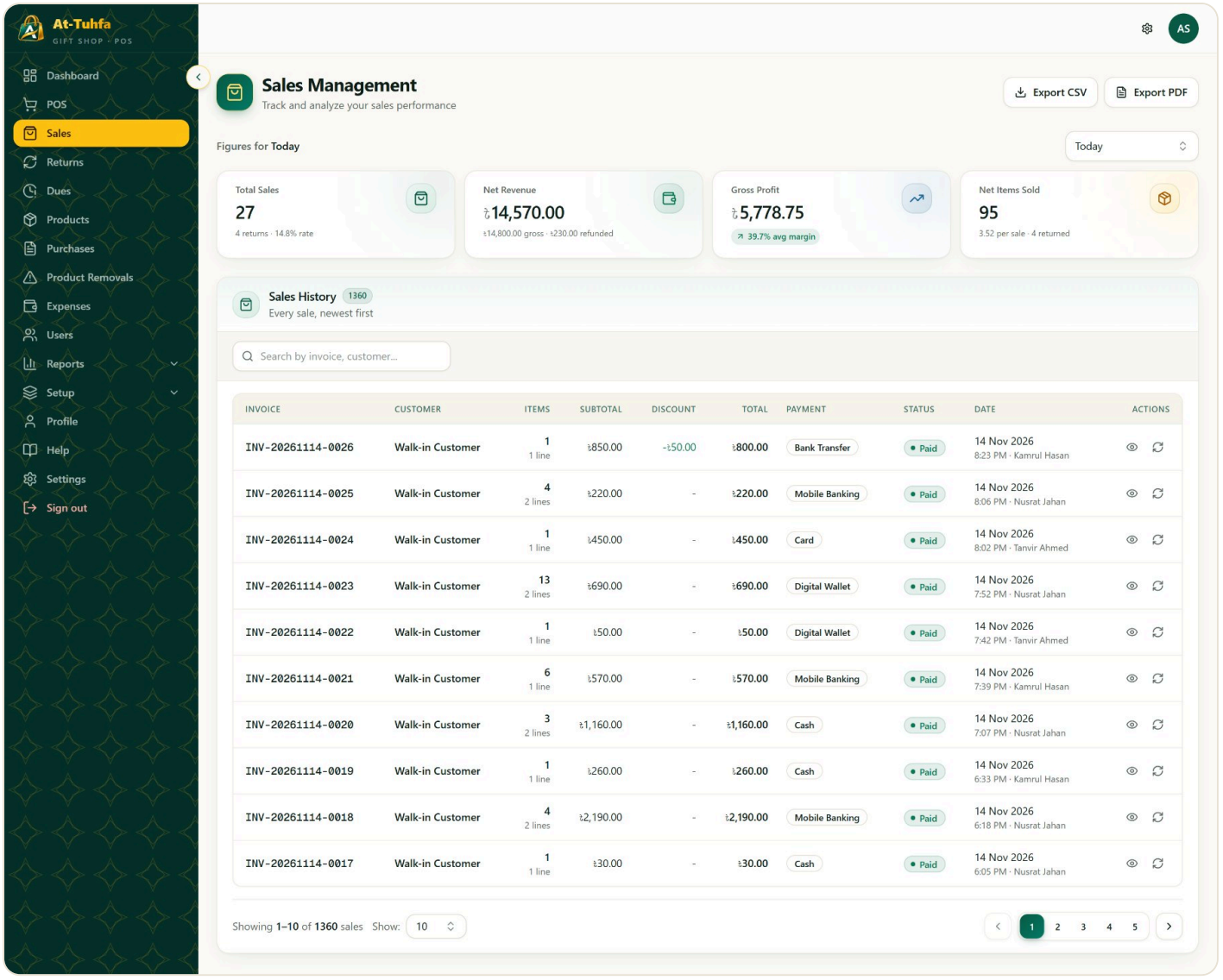
One screen answers "how are we doing". Revenue, gross profit, net profit and money owed sit across the top, and a period filter drives everything beneath them: a revenue trend, a profit chain that walks from takings down to what the shop kept, the products actually earning their shelf space, and a list of what is running low, ready for the next supplier order.



Shop overview

Sales history

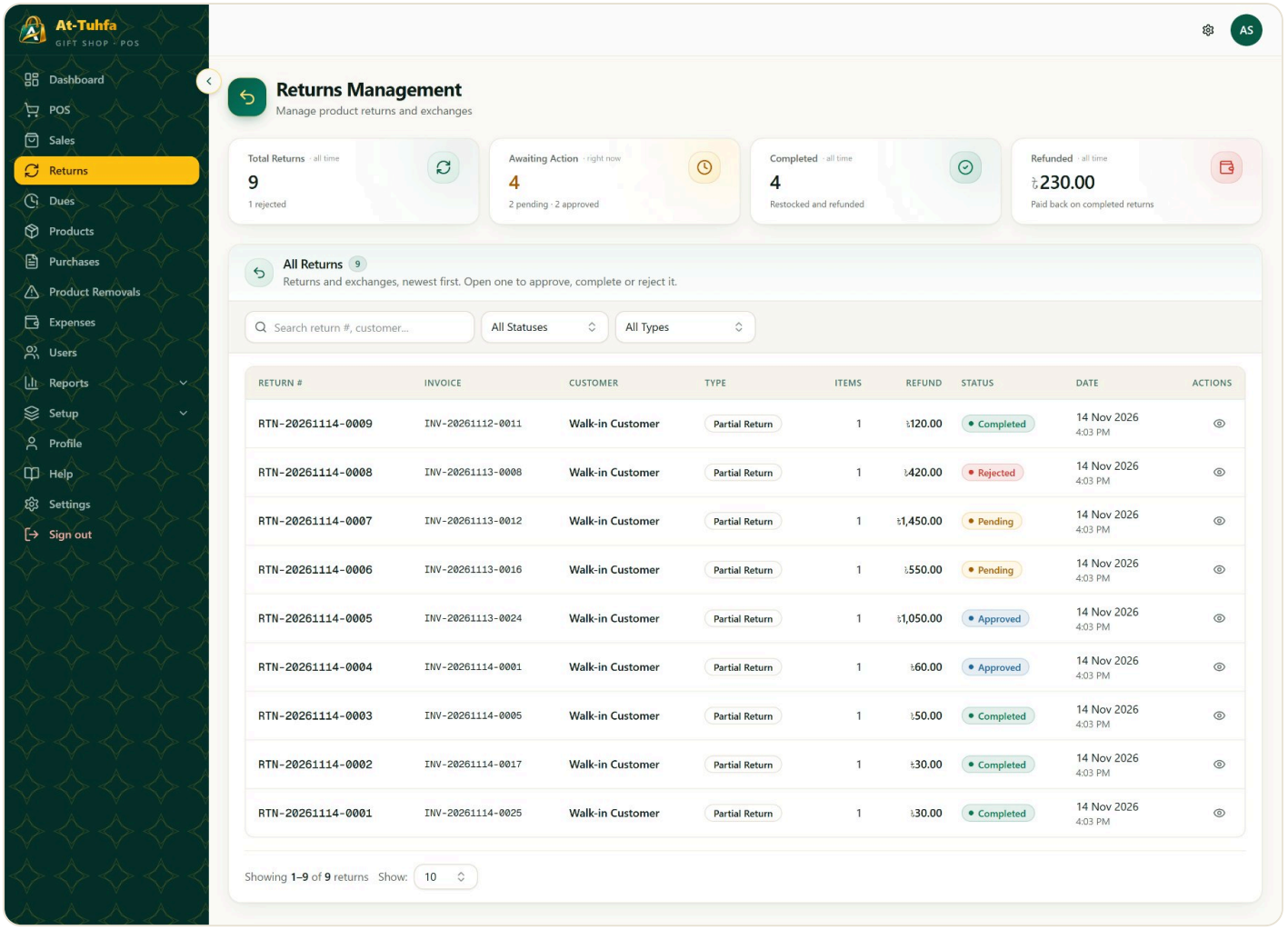
Every invoice is searchable by number or customer, filterable by period, and exportable to CSV or a branded PDF that carries the shop details from Settings. Opening a sale shows the line items, what was paid, what is still owed and anything already returned against it. Returns start here, from the row of the sale being returned.



Sales

Returns and exchanges that cannot leak

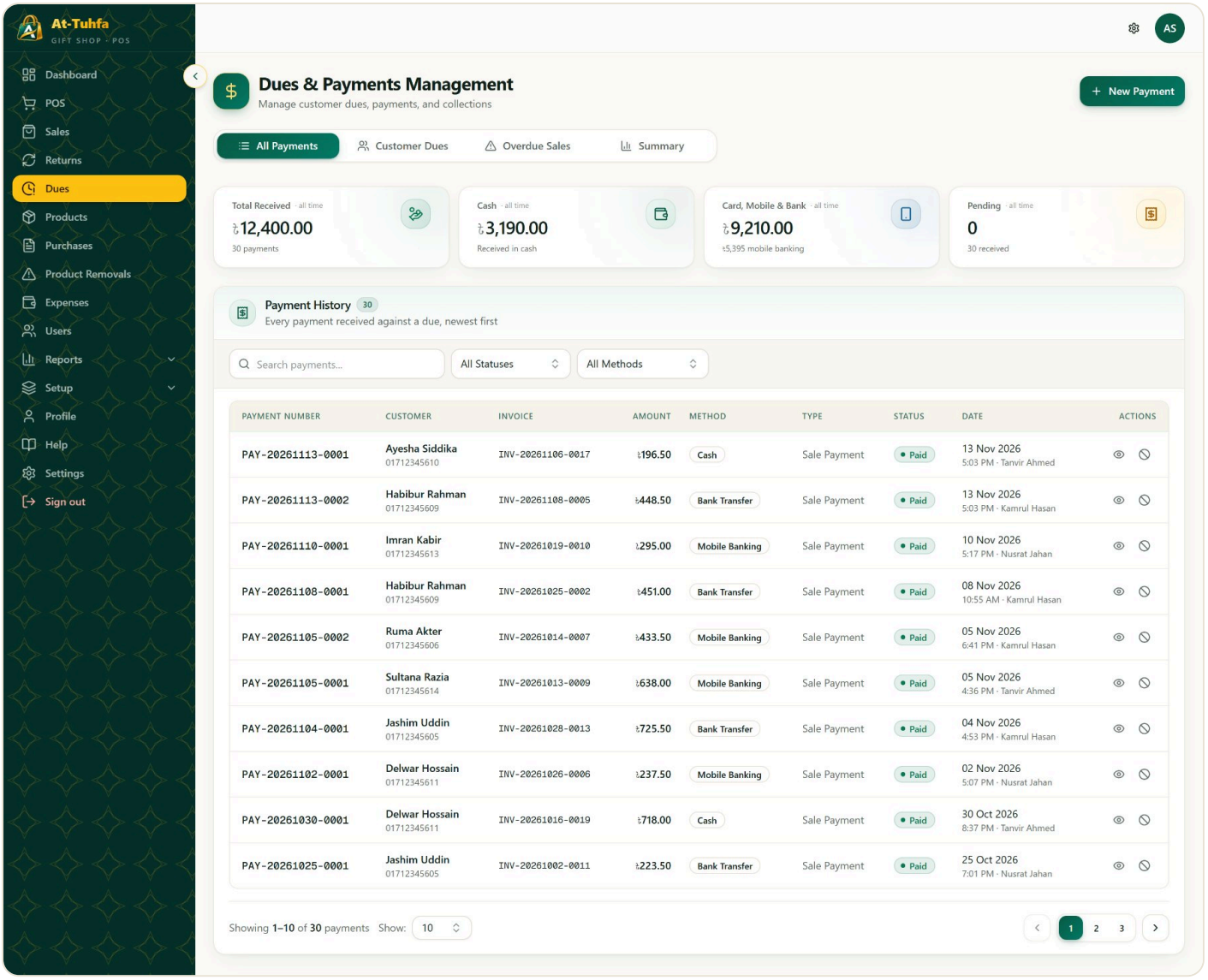
A return is a three-step workflow, not a button: `PENDING → APPROVED → COMPLETED`. **Stock only re-enters inventory on the final step**, so goods a manager has not yet accepted never quietly inflate what the shop thinks it owns. Refunds are calculated at the price actually paid on the original sale, not today's price, and a restocking fee can be withheld. An exchange follows the same path and hands out the replacement product when it completes.



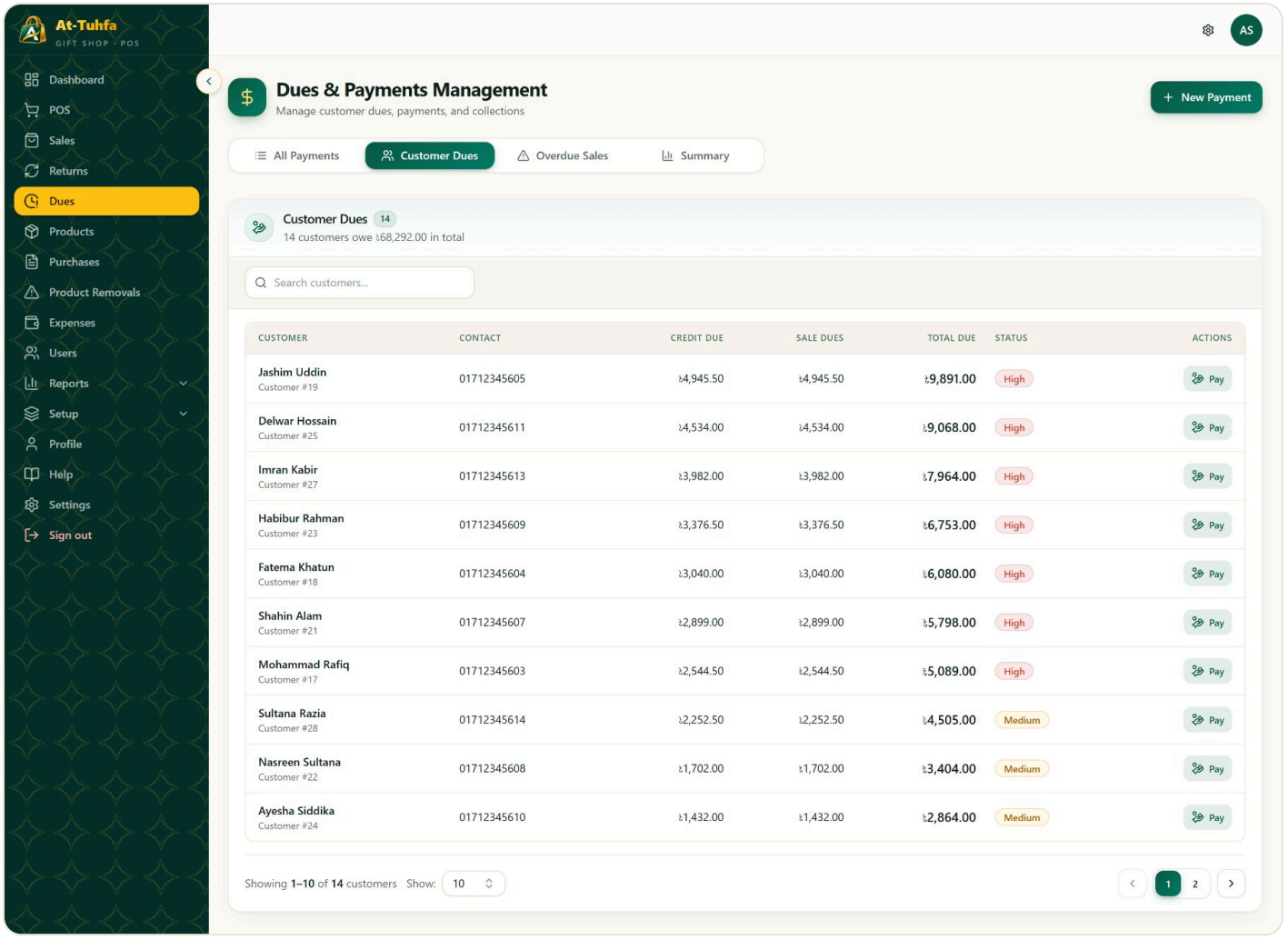
Returns

Credit, and getting paid

Selling on credit is normal here, and it is where shops quietly lose money. A customer is created automatically on their first credit sale, keyed on phone number. Four tabs cover the whole cycle: every payment taken, who owes what, which invoices are overdue, and a summary that flags customers past their credit limit.

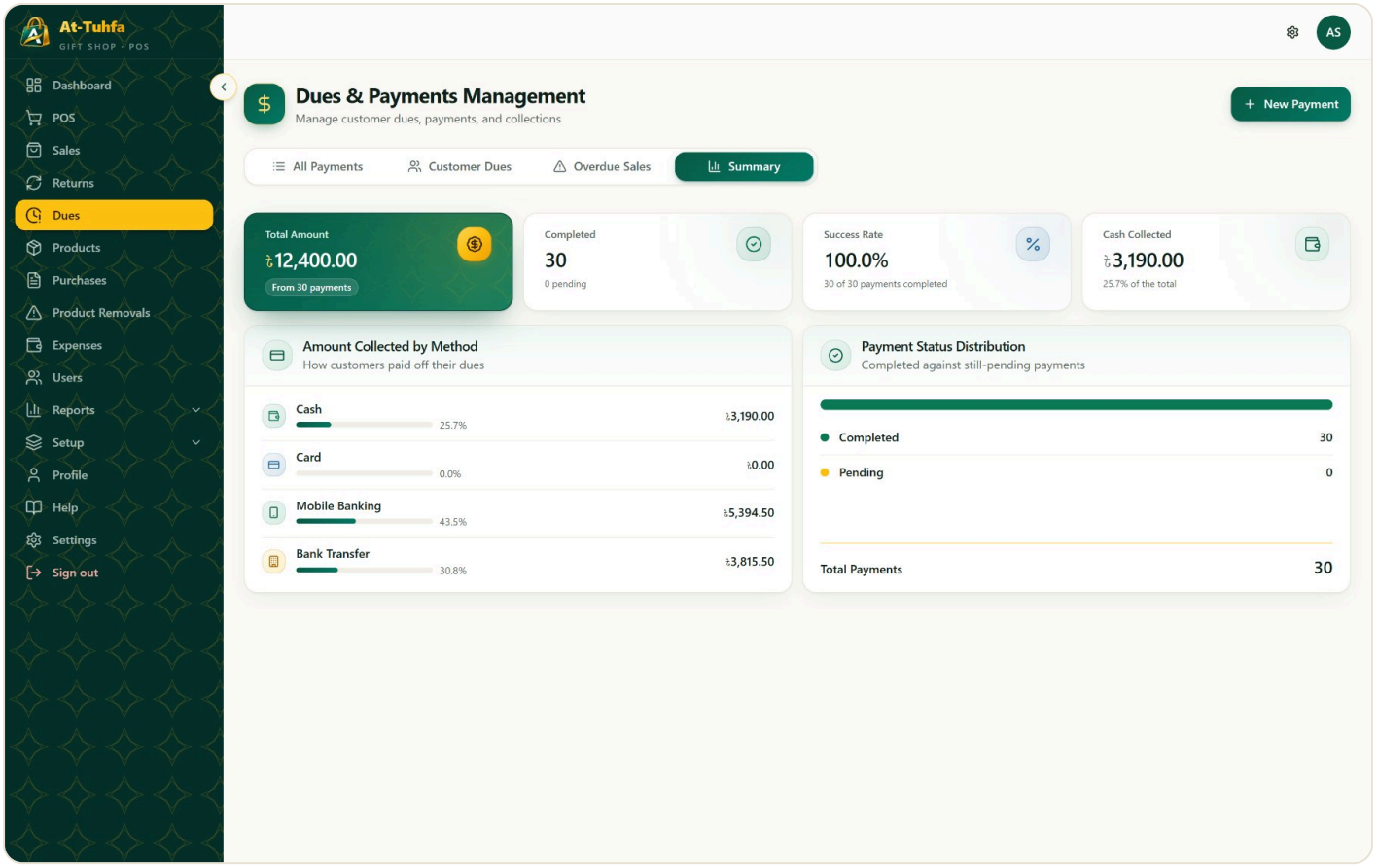


Dues and payments



Who owes what

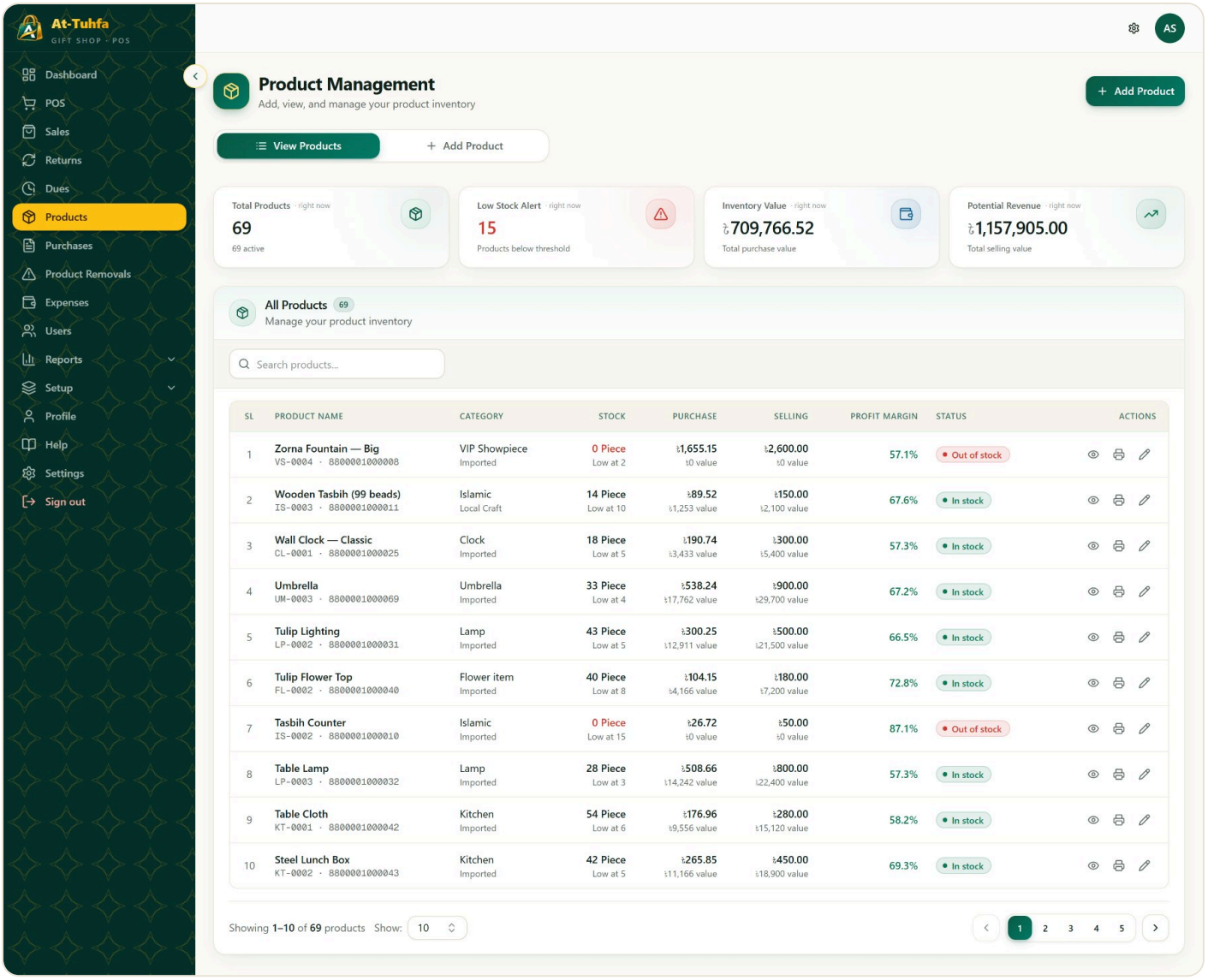
One payment can be split across several invoices, and each method asks for the reference that matters: a bKash transaction ID, a bank account or a cheque number. Every movement writes a permanent ledger line recording the balance before, the change and the balance after.



The credit summary

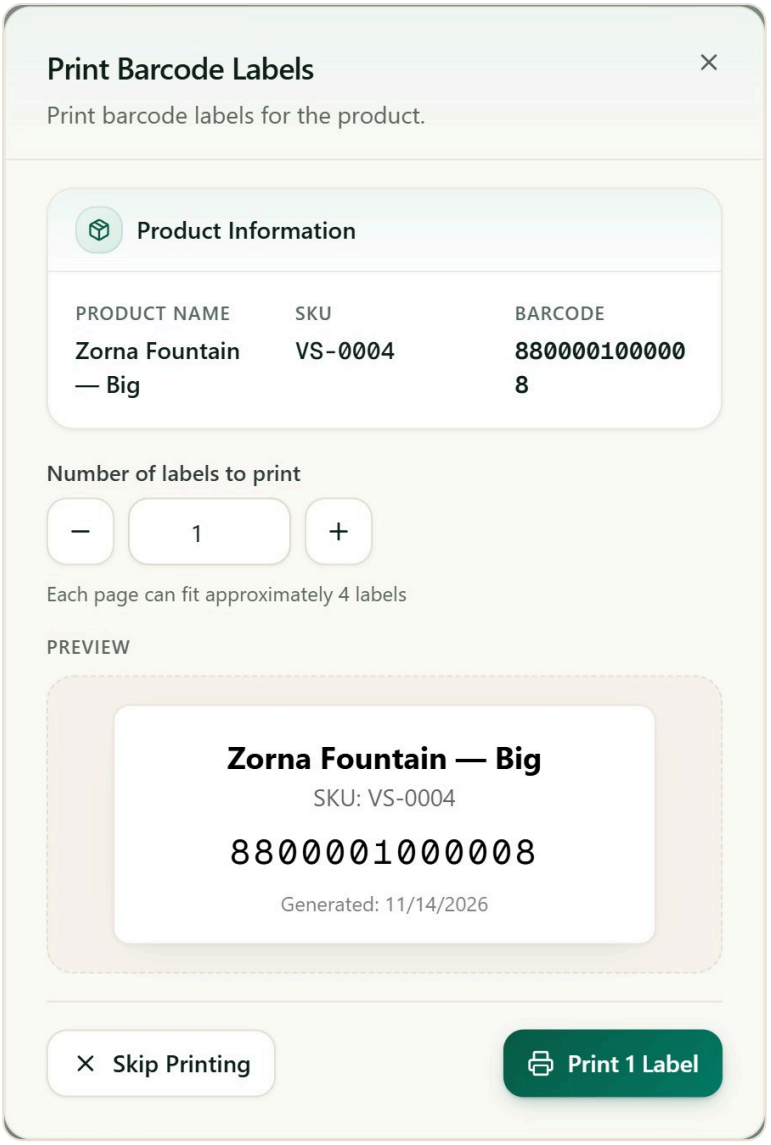
Products and barcodes

The catalogue carries SKU, barcode, category, brand, unit, cost, price, shelf location and a low-stock threshold. SKUs generate automatically or can be typed by hand. A barcode is issued on save and is guaranteed unique, or a manufacturer's barcode can be used instead.



Products

Labels print straight from the product list, as many as the shelf needs.

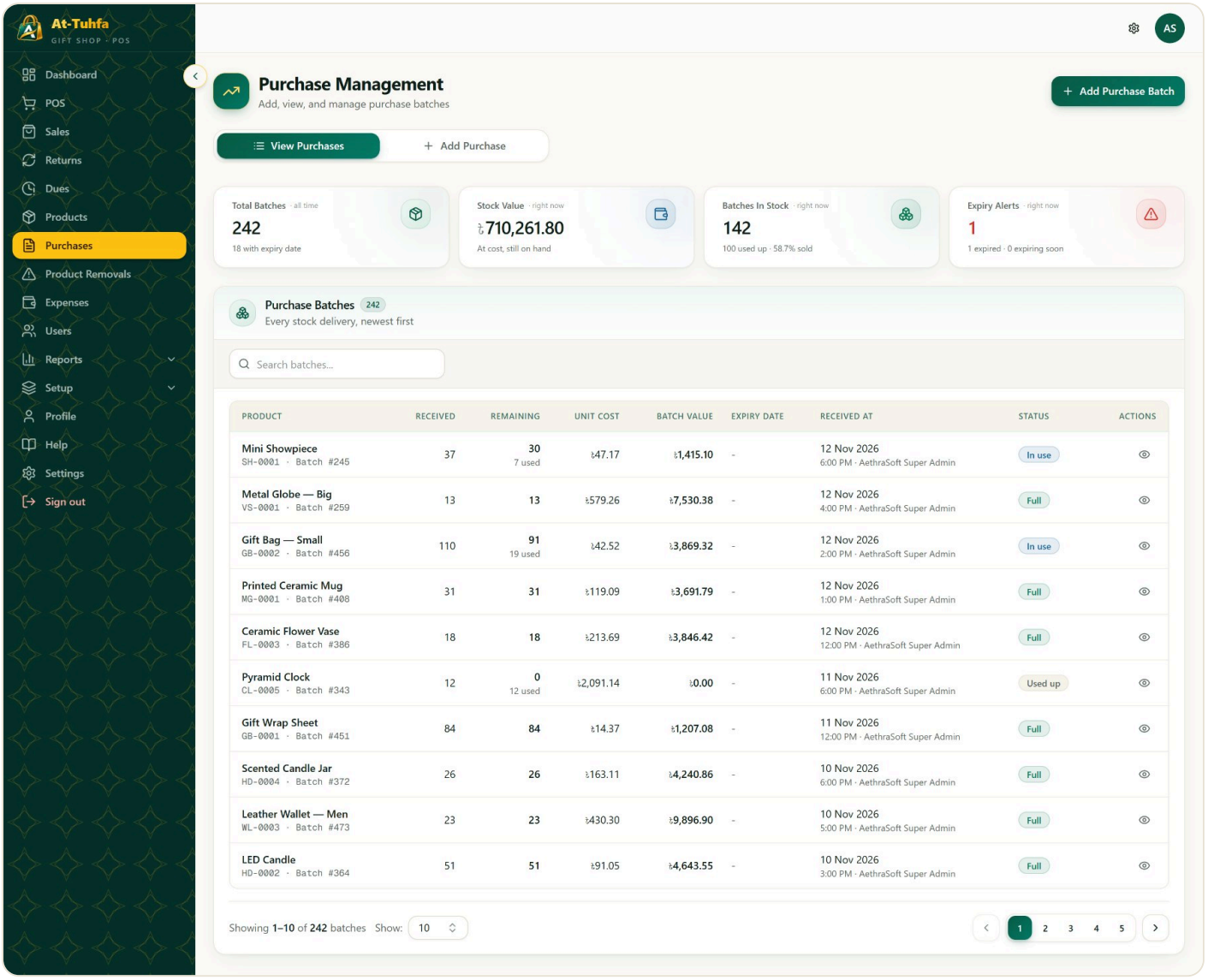


Printing barcode labels

Stock quantity is deliberately **not** editable here. Stock changes through purchases, sales, returns and removals, never by someone typing over it, which is the only way the cost trail survives.

Receiving stock

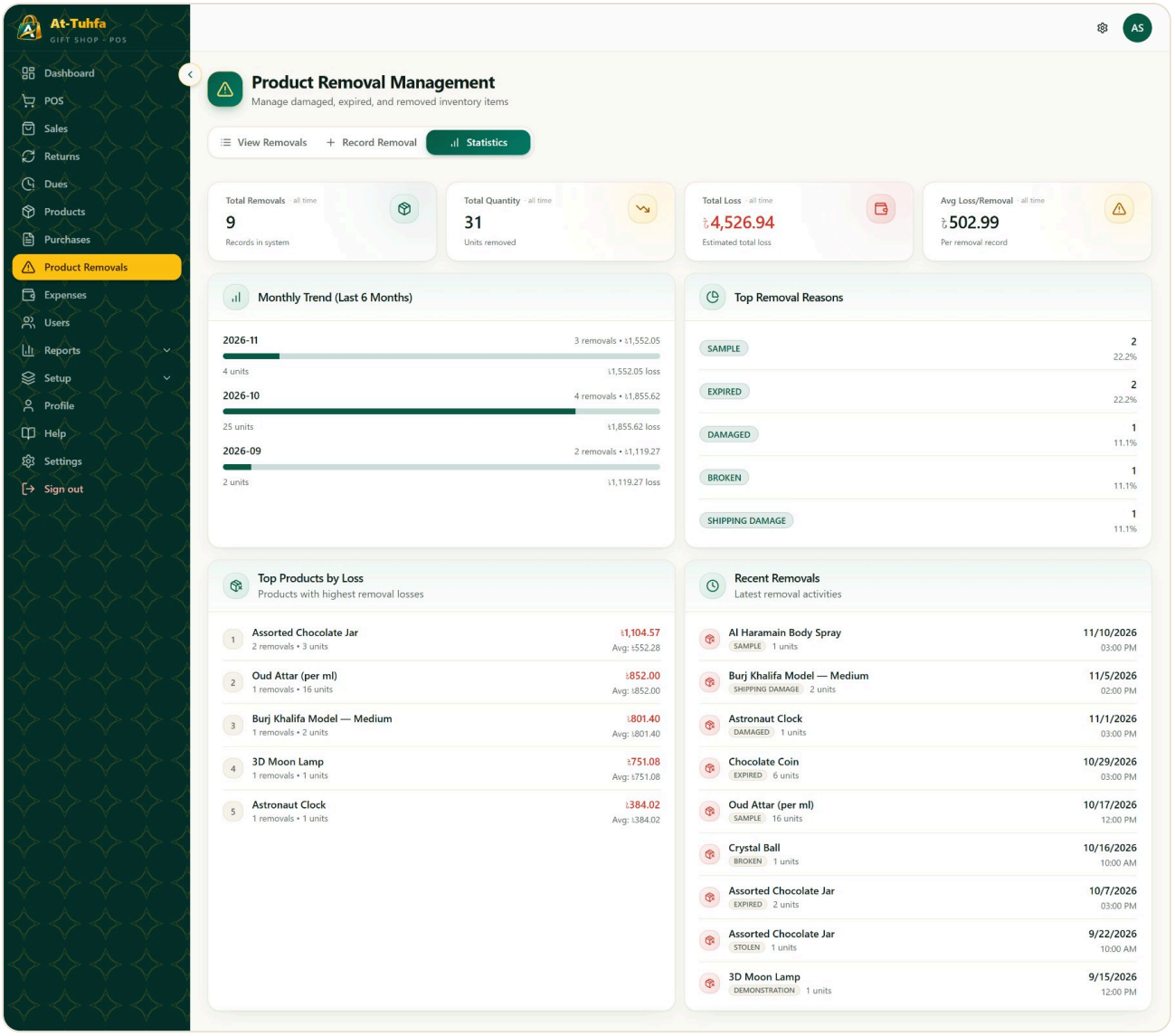
Stock arrives as batches: a quantity, a unit cost, a date and, for anything perishable such as chocolate, an expiry date. Sales draw from batches oldest-first, so old stock clears before new. Each receipt recalculates the product's weighted-average cost, and every sale line freezes that cost at the moment of sale. The statistics strip shows stock value at cost, batch use and expiry alerts.



Purchase batches

What the shop loses

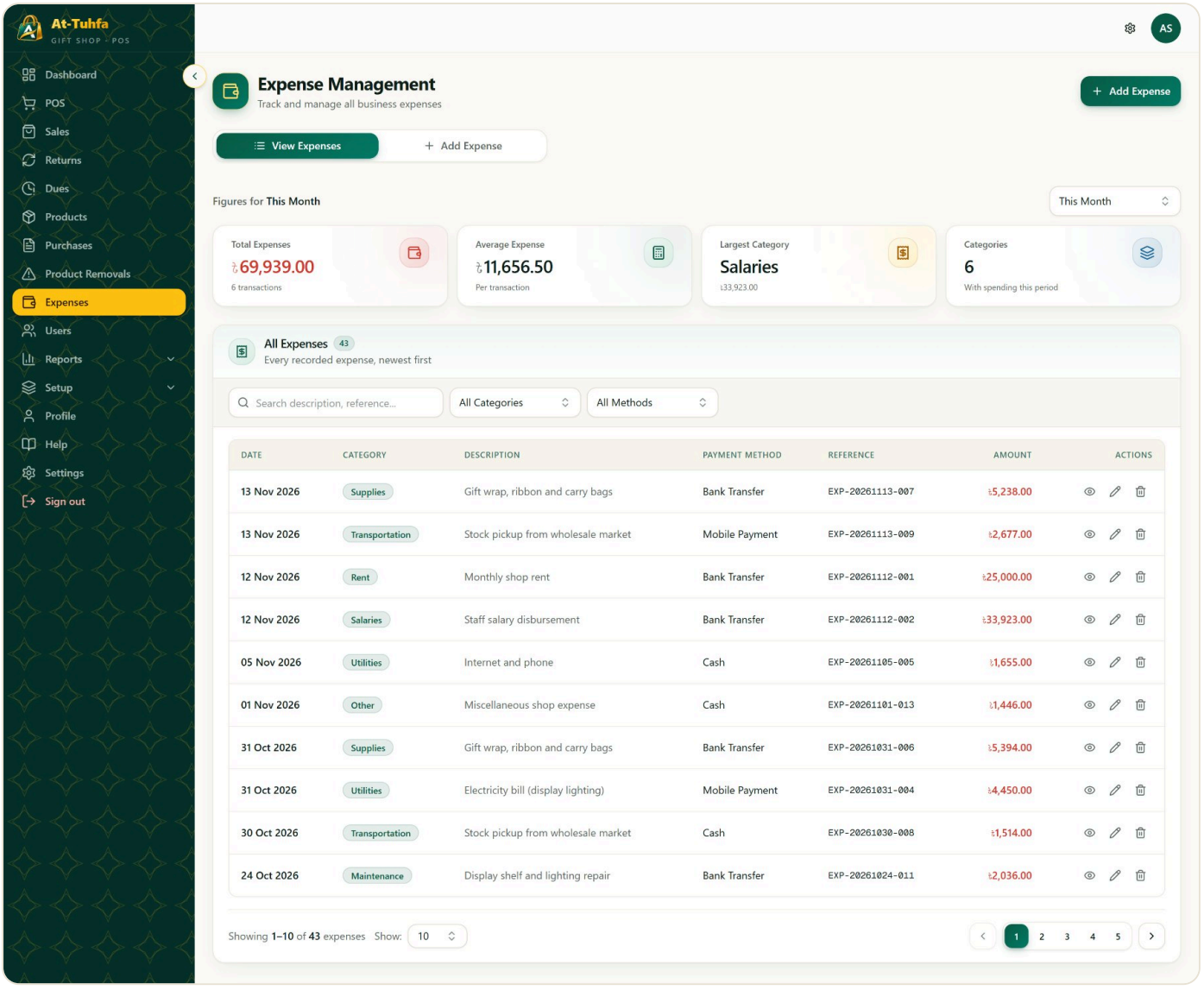
Broken showpieces, expired chocolate, attar given away as testers, stock that walked out of the door: twenty-one reasons, each with an estimated loss. Removals deduct stock immediately with no approval step, deliberately unlike returns, because the goods are already gone. The Statistics tab turns the shrinkage into something you can act on.



Removal statistics

Overheads

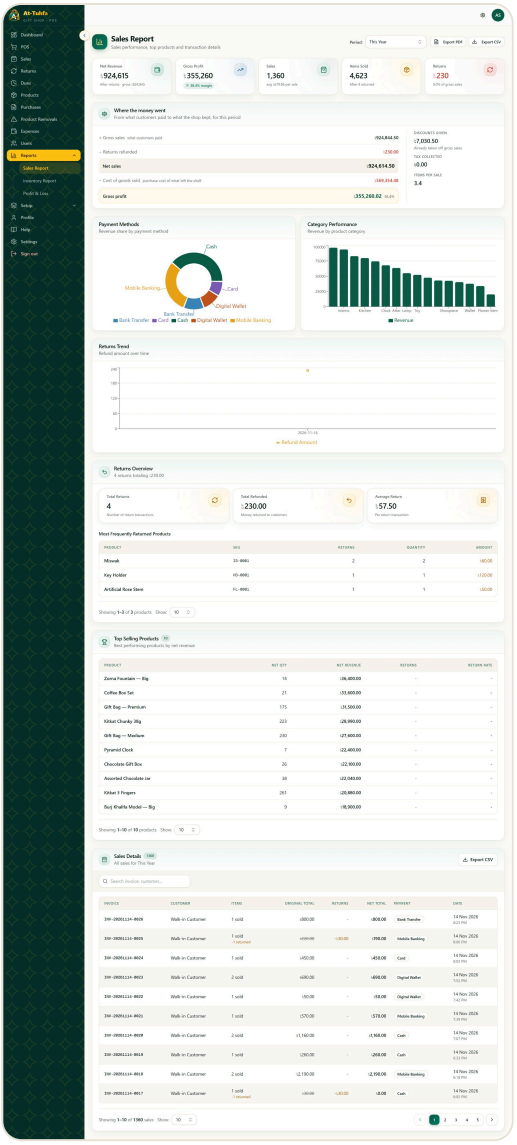
Rent, salaries, utilities, packaging, transport and promotion: ten categories and five payment methods, and a receipt image can be attached to any of them. These are what turn gross profit into the number that matters.



Expenses

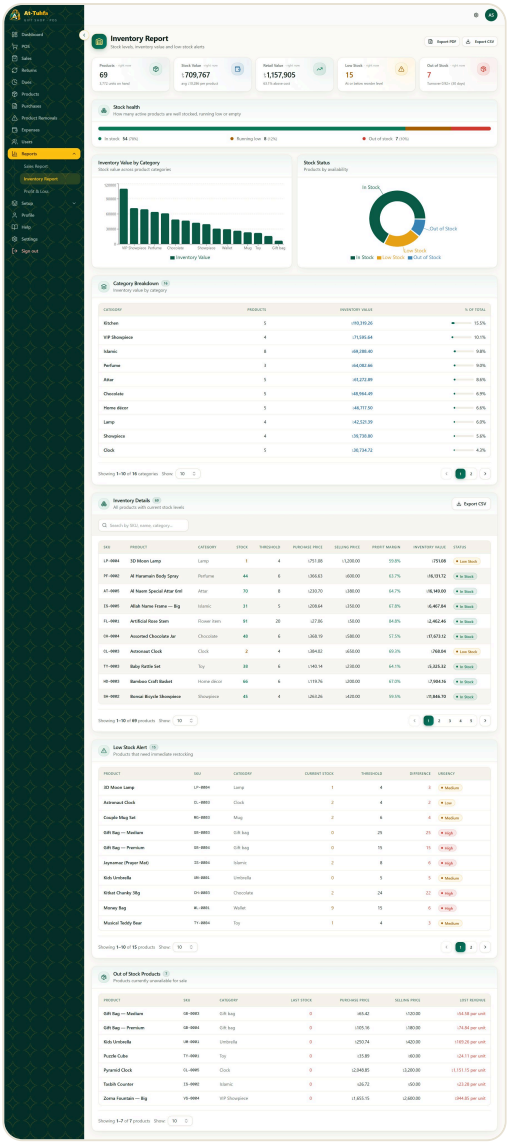
Three reports

Sales covers gross sales, returns, net revenue, net profit, average order value and return rate, with payment-method and category breakdowns and a full transaction table.



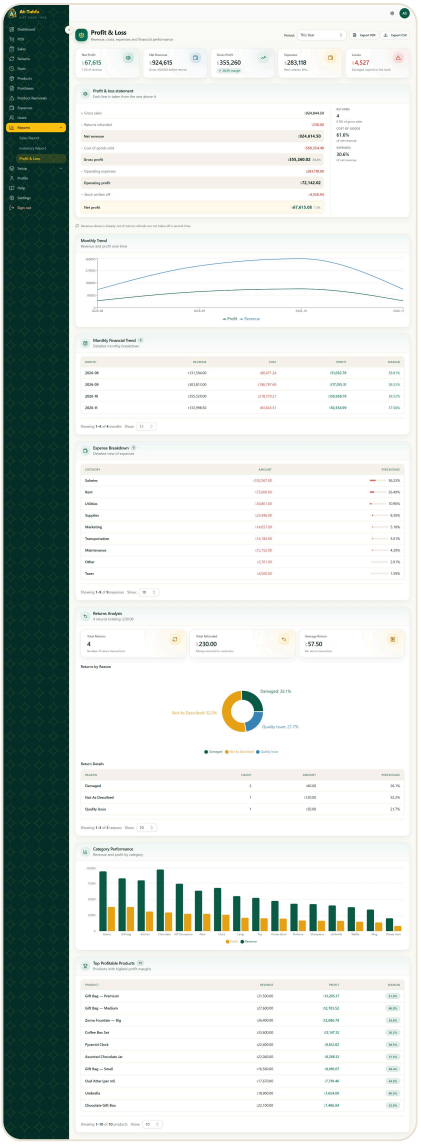
Sales report

Inventory is always live and never period-filtered. It shows stock at cost, stock at retail and per-product margin, plus the two lists that drive purchasing: low stock and out of stock.



Inventory report

Profit & Loss is the whole statement, from gross sales down to net profit, with every line traceable to the screen it came from.

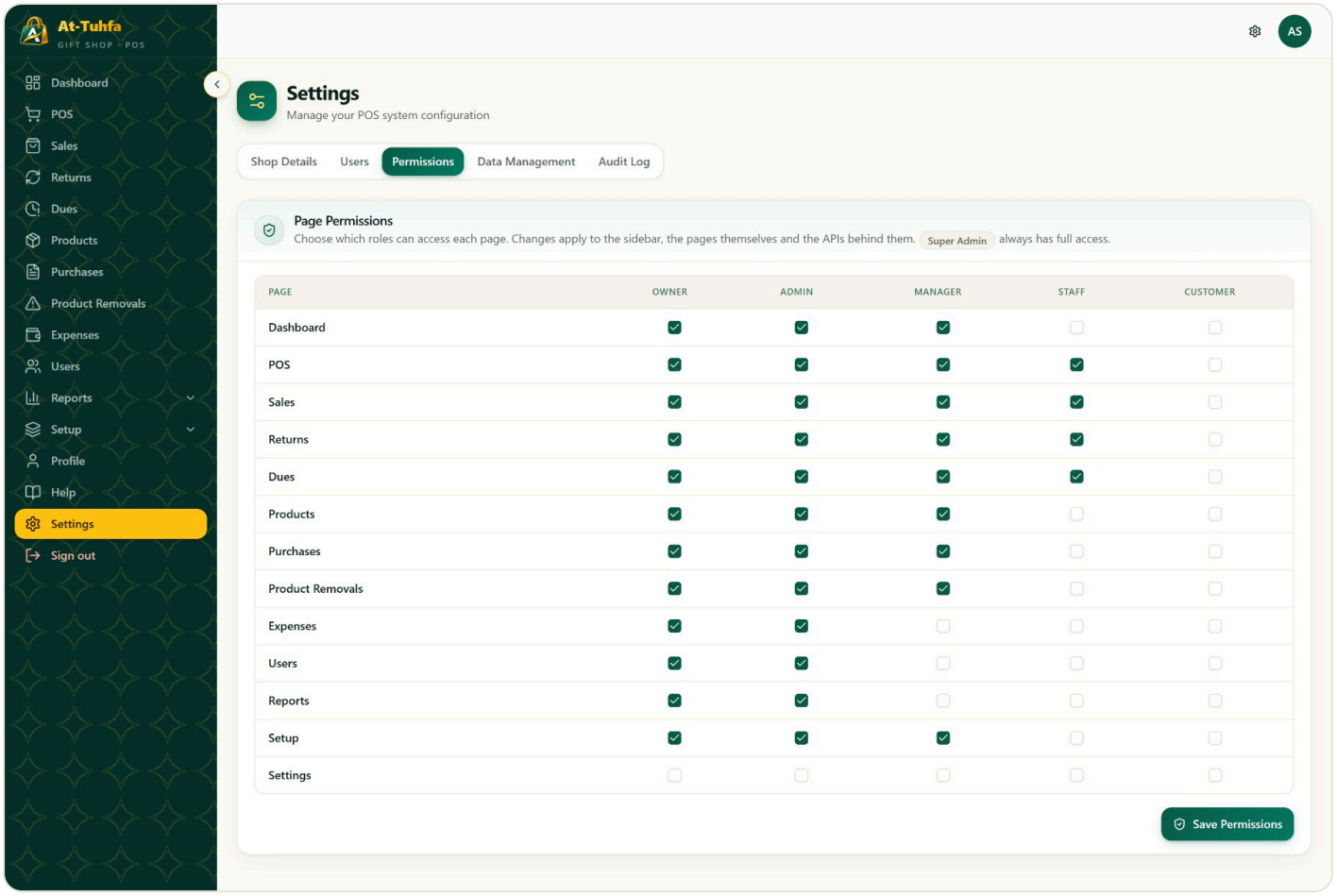


Profit and loss

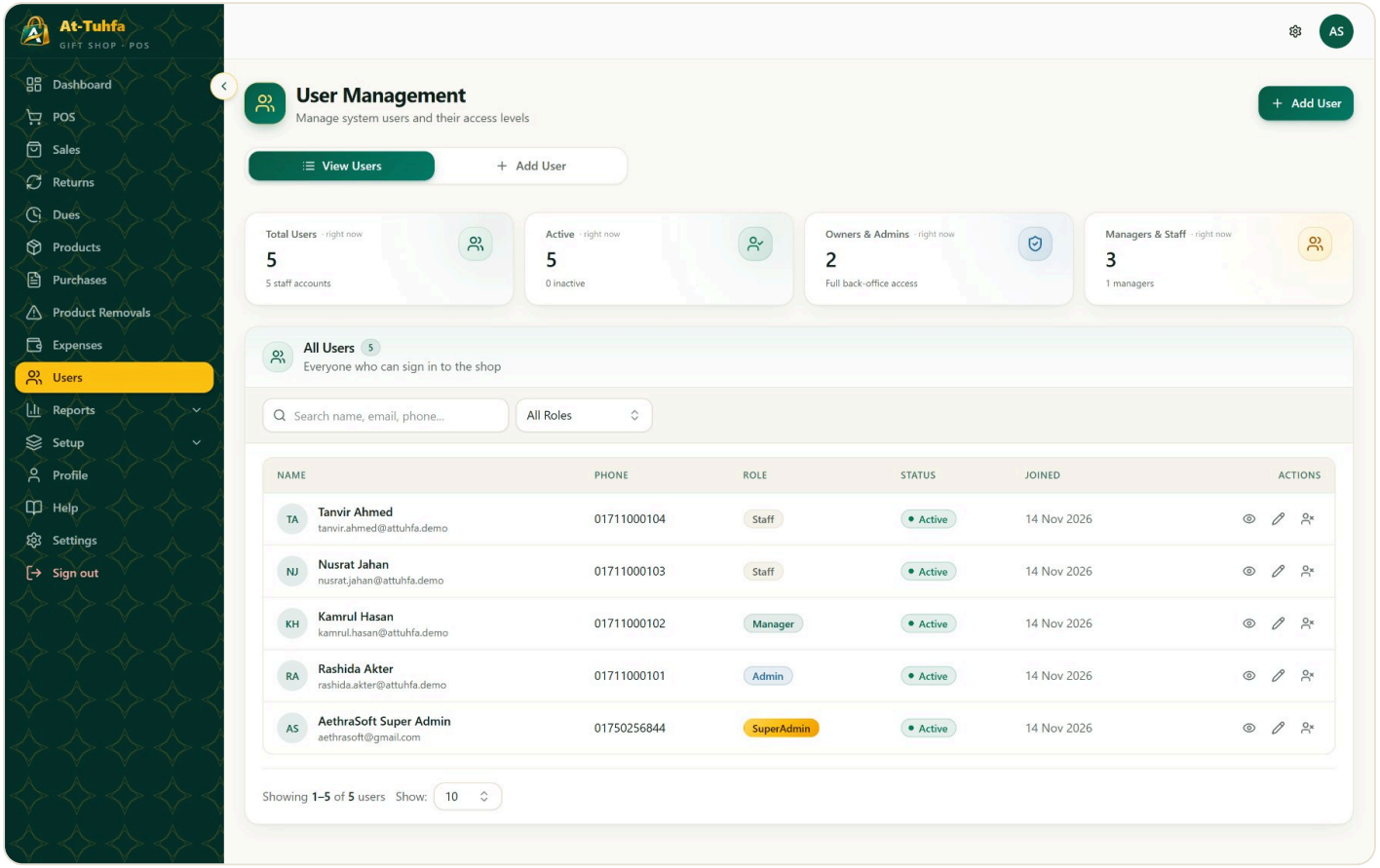
All three export to CSV and to a branded PDF carrying the shop's details. Days are bucketed on Asia/Dhaka time, whatever the server's clock is set to.

Who can see what

There are six roles: SuperAdmin, Owner, Admin, Manager, Staff and Customer. Access is not compiled in. The SuperAdmin edits a page-by-role grid in Settings, and it takes effect on the user's next navigation, in the sidebar, on the page and on the API behind it.



The permission grid

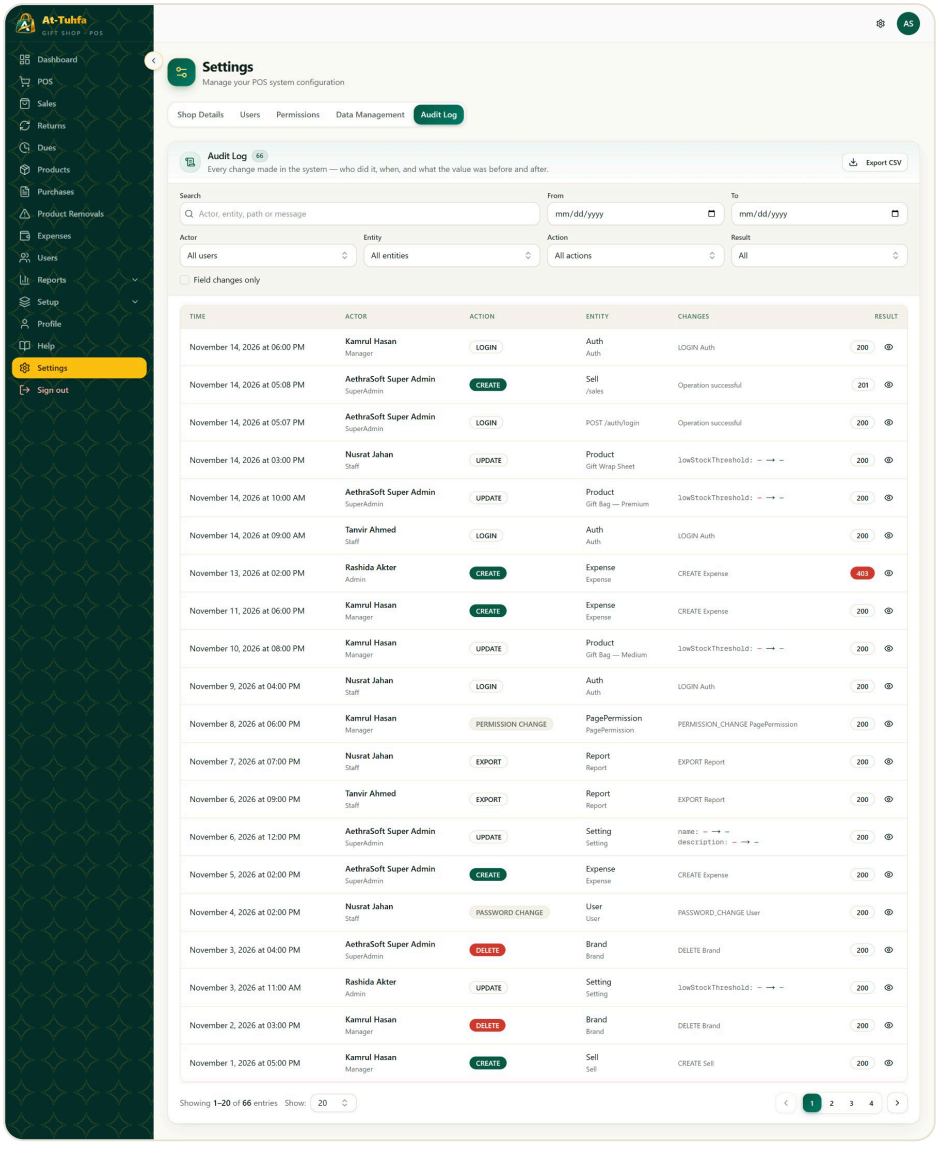


Users

Staff land on the POS after signing in. Everyone else lands on the dashboard.

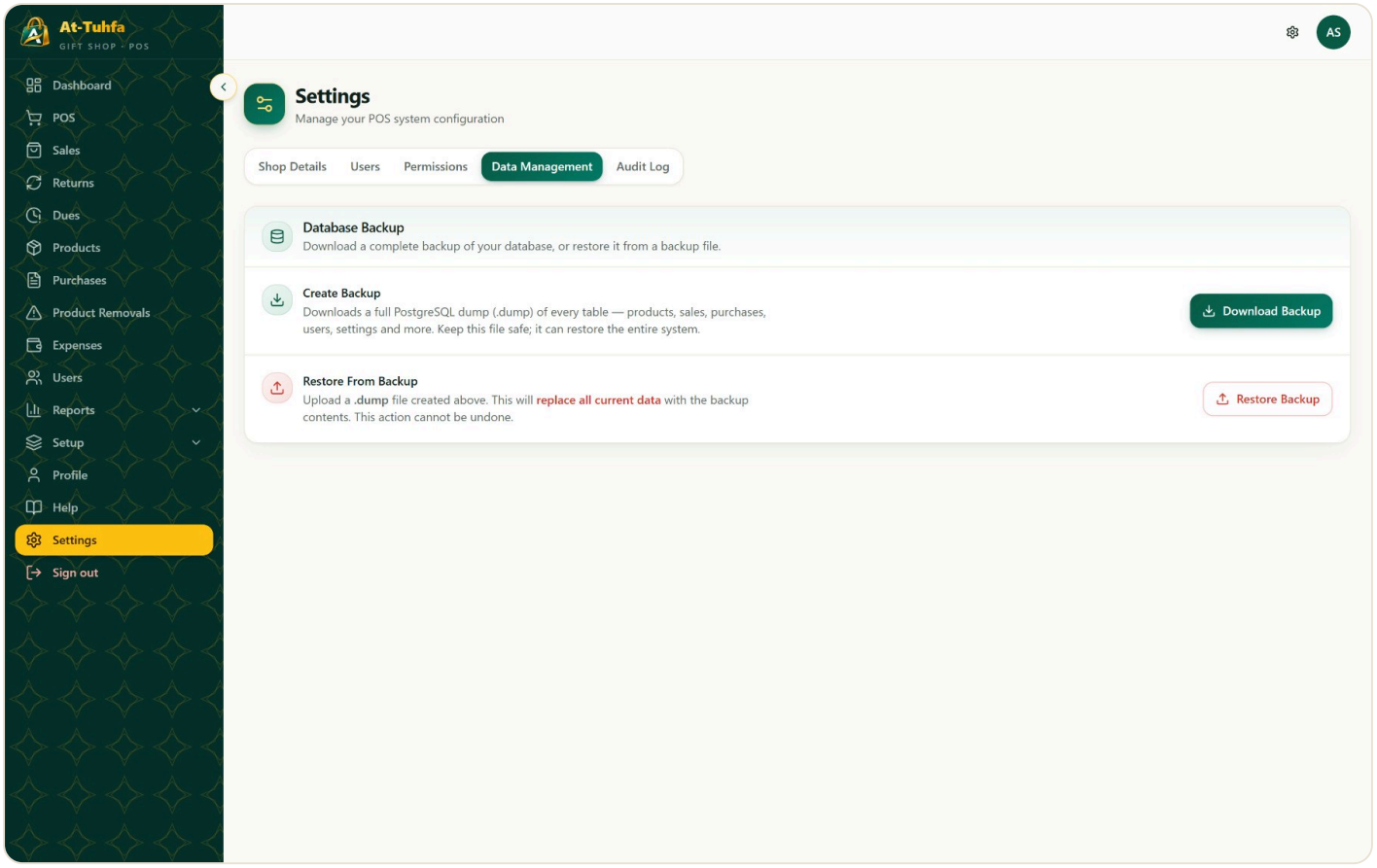
The paper trail

Every request that changes data is recorded: who, what, when, from which address, and the fields that changed, with their values before and after. Nobody can edit or delete the log, the SuperAdmin included, and the whole filtered set exports to CSV.



Audit log

Next to it, a one-click PostgreSQL backup downloads the entire database, and a restore puts it back.



Backup and restore

Under the hood

Frontend: Next.js 16 (App Router) with React 19 and TypeScript. Tailwind CSS v4 with a shadcn/ui component set over Radix primitives. TanStack Query for server state, Redux Toolkit for session state, React Hook Form with Zod for every form and Recharts for the charts. JWT access and refresh tokens are held in cookies and rotated inside Next middleware.

Backend: NestJS 11 over PostgreSQL through Prisma 6, in sixteen service modules. Passport JWT and bcrypt for auth, class-validator for request contracts and Swagger for the API surface. Helmet and a rate limiter sit at the edge, with Winston and daily log rotation, Multer for uploads and a Terminus health endpoint that reports on the database.

```
Browser ──▶ one Node process on one port
            ├─ Next.js 16   storefront, back office, POS
            └─ NestJS 11    /tohfa/api/v1 ──▶ Prisma ──▶ PostgreSQL
                            guards, page permissions,     batches, le
                            audit interceptor             audit trail
```

Decisions worth pointing at

One process, one port. NestJS mounts Next.js as its first middleware, ahead of its own body parser, so Server Actions keep working and the whole product deploys as a single service.

Permissions live in the database, not the code. A `page_permissions` table maps each page to the roles allowed to open it. One source feeds the sidebar, the route guard and the API guard, so a permission change cannot leave the menu and the API disagreeing. SuperAdmin is deliberately absent from the grid, so you cannot lock yourself out.

Stock is a number with cost layers underneath. A product's stock figure is authoritative. Purchase batches are the cost layers under it, consumed oldest-first. Cost of goods sold is the weighted-average cost, frozen on each sale line at the moment of sale, so a report run next year still shows what that sale really earned.

Every stock change locks its rows first. A sale, return or removal runs in one database transaction and locks the affected products before touching them, so two tills selling the last item at the same moment cannot both succeed.

Document numbers come from a sequence, not a count. `INV-`, `PAY-` and `RTN-` numbers are drawn inside the same transaction that inserts the document, never derived from a row count, so two sales completed at the same moment can never share a number.

The shop's day is Dhaka's day. Timestamps are stored in UTC, but "today", "this month" and every daily bucket in the reports follow Asia/Dhaka, on the server and in the browser.

The ledger is append-only. Customer balances are never simply overwritten. Each movement records the previous balance, the change and the new balance, so a disputed due can be walked back line by line.

Documentation is generated, not maintained twice. Fourteen chapters in English and Bangla live as Markdown. A dependency-free build script compiles them into two self-contained HTML manuals with client-side search, scroll-spy navigation and a print stylesheet that turns "Save as PDF" into a real book.

The screenshots on this page are generated too. A script seeds the demonstration shop, signs in, walks every screen, opens the dialogs and tabs that matter, completes a sale to capture the receipt and writes the PNGs at 2× resolution. The app is a fixed-height shell that scrolls internally, so a naive full-page capture returns only the viewport. The script measures each page's scroll container and grows the viewport to fit instead.

Deliberately not built

Being straight about the edges is more useful than pretending there aren't any. As it stands, the system does not do these:

- Online ordering: the storefront has no cart, payment or delivery, and its content is edited in one data file rather than pulled from the POS catalogue
- Tax on a sale: a tax rate is stored in Settings but not applied at checkout
- Reprinting a receipt: it prints once, when the sale completes
- A receipt header from Settings: the PDF exports use the shop details from Settings, but the receipt header is still fixed in code
- Self-service password reset: an administrator sets passwords

---

Built by AethraSoft

AethraSoft did the design, frontend, backend, database and documentation.

- Storefront: attuhfa.mahfuz.tech
- POS demo: pos.mahfuzislam.tech
- Company: aethrasoft.com
- Email: aethrasoft@gmail.com
- Phone: 01750256844
- GitHub: github.com/Mahfuzislam1695
- Web: mahfuzislam.tech

Made in Bangladesh.